CODE PATH *ORG

Dictionaries and Sets

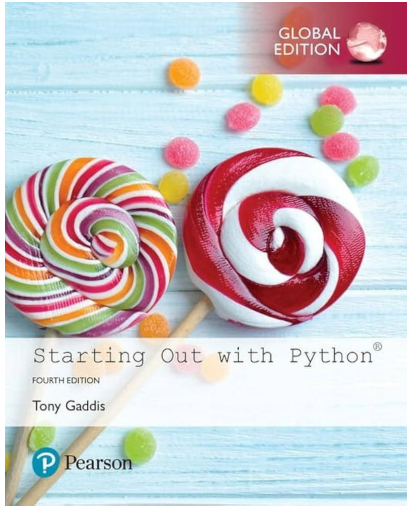
CIIC 3015 - Lecture 06

Karelys López Rivera

Lecture Goals

- **Dictionaries**
  - What are they?
  - Why do we need them?
  - How to create and use them?

- **Sets**
  - What are they?
  - Why do we need them?
  - How to create and use them?



Starting Out With Python (4th Edition) Chapter 9

Recap

Lists

- Create list: putting comma-separated values inside of square brackets []
- Lists are mutable
- Lists are a sequence of elements
- Can contain zero or more values of any type
    - Usually same type in one list
- Can contain multiple types, but rarely useful
- Examples:

```
ages = [18, 26, 19, 21, 23, 34]
states = ["AL", "MS", "SC", "CA"]
answers = [True, True, False, True]
weird = [True, 1, 1.0, '1']
```

Lists

- Accessing elements in a list by index:
  - If using positive numbers: starts at 0 and the last valid index will be the length - 1
  - If using negative numbers: starts at -1 and the last valid index will be -length (negative length)
  - Examples:
    - my_list = [1 , 2, 3, 4, 5, 6, 7, 8, 9, 10]
    - my_list[0] -> 1
    - my_list[len(my_list)-1] -> 10
    - my_list[-len(my_list)] -> 1
    - my_list[-1] -> 10

Lists

- Slicing
  - my_list = [1 , 2, 3, 4, 5, 6, 7, 8, 9, 10]
  - my_list[:4] -> [1, 2, 3, 4]
  - my_list[4:] -> [5, 6, 7, 8, 9, 10]
  - my_list[1:4] -> [2, 3, 4]
- Modifying values in list
  - my_list = [1 , 2, 3, 4, 5, 6, 7, 8, 9, 10]
  - my_list[0] = 10 -> my_list = [10 , 2, 3, 4, 5, 6, 7, 8, 9, 10]

Lists

- Loop in List

my_list = [1 , 2, 3, 4, 5, 6, 7, 8, 9, 10]		
for loops	for loops with range	while

for num in my_list:
  print(num)
	for i in range(len(my_list)):
	  print(my_list[i])
		i = 0
		while i < len(my_list):
		  print(my_list[i])
		  i+=1

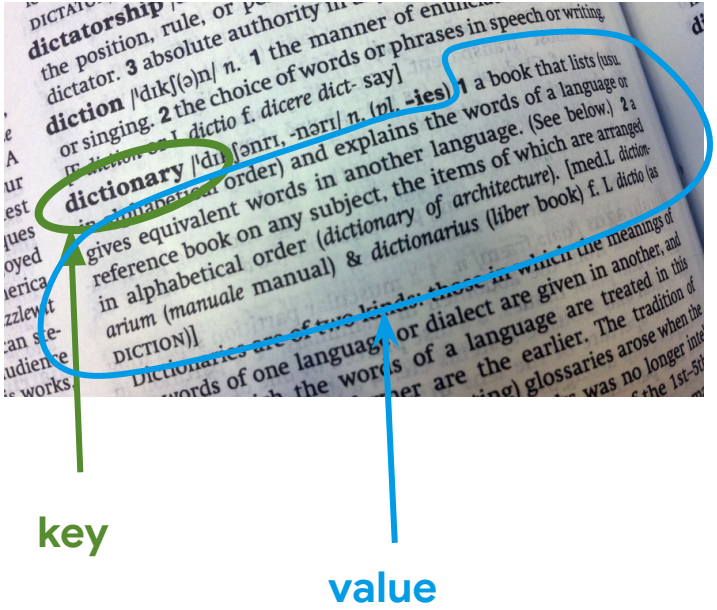
Strings

- Strings are immutable
- Sequence of characters
- A string can be empty (“”) or contain characters (“hello!”)
- Just like lists you can use len() to know how many characters does a string have)
- You cannot change string directly once they’re created.
    a. Example: word = “cat” and you want word to be “cut you cannot modify a string directly
    b. new_word = cut
    c. word = new_word
- Strings can also use indexing, slicing and loops like lists! Why? Because they are sequence 😁

Dictionaries

Real Life Examples

- UPR email accounts
- UPRM student ID
- Instagram username
- dictionary


dictionary
gives equivalent words in another language. (See below.)
reference book on any subject, the items of which are arranged
in alphabetical order (dictionary of architecture).
key
value

Introduction: Dictionaries

- A **dictionary** is a labeled collection of data items.
  - A dictionary stores data in pairs of keys and values.
  - A key is a unique label associated with a specific value.
  - You can use a key to retrieve its corresponding value.

```
inventory = {"cheese": 10, "milk": 37, "yogurt": 6}
```

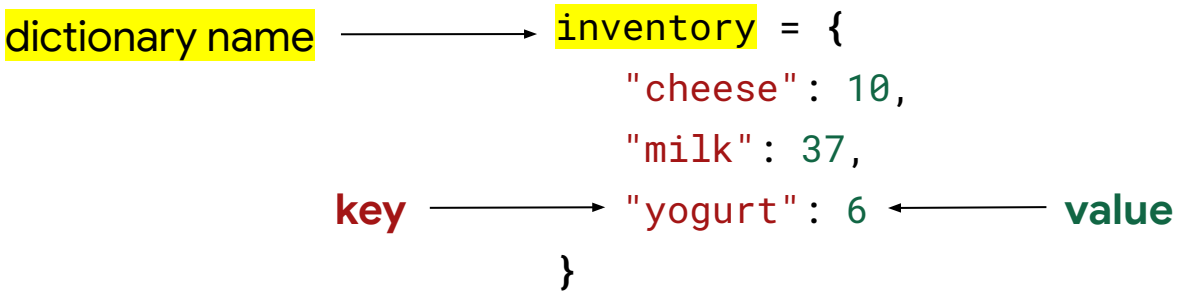
Keys and values

- Data types
  - A **key** must be an immutable data type
    - ints, floats, bools, and str are all ok to be keys.✅
    - A list, set, or dictionary cannot be a key in a dictionary. ❌
  - A **value** may be any data type
- Uniqueness
  - All **keys** in the dictionary must be **unique**
  - Different dictionary items can have the same value

Syntax: Create a dictionary

The general syntax for creating a dictionary:

1. Start with the dictionary name followed by the assignment operator (=)
2. Use curly braces ({}) to enclose the contents of the dictionary
3. Use colons (:) to separate keys from their corresponding values
4. Use commas (,) to separate **key**-**value** pairs from each other



Syntax: Create a dictionary, another way!

The general syntax for creating a dictionary:

1. Start with the dictionary name followed by the assignment operator (=)
2. Use dict() to initiate a dictionary

dictionary name → `inventory = dict()`

Examine the code

- What are the keys in the `student_grades` dictionary?

```
student_grades = {
    "calculus": 92,
    "biology": 88,
    "computer science": 95
}
```

Examine the code

- What are the keys in the `student_grades` dictionary?

```
student_grades = {
    "calculus": 92,
    "biology": 88,
    "computer science": 95
}
```

- The keys in the `student_grades` dictionary are `"calculus"`, `"biology"`, and `"computer science"`.

Display a dictionary

- Use a print statement to display all the key-value pairs in a dictionary.

```
inventory = {
    "cheese": 10,
    "milk": 37,
    "yogurt": 6
}

print(inventory)
```

```
{'cheese': 10, 'milk': 37, 'yogurt': 6}
```

Access a dictionary value

- Access a dictionary value using its key with square brackets [ ]

```
inventory = {
    "cheese": 10,
    "milk": 37,
    "yogurt": 6
}

print(inventory["cheese"])

10
```

Check your understanding

- What is the output of this code block?

```
customer = {
    "name": "Avery",
    "id": 4195,
    "purchases": [75.25,110.34],
    "rewards": False
}

print(customer["purchases"])
```

Check your understanding

- What is the output of this code block?

```
customer = {
    "name": "Avery",
    "id": 4195,
    "purchases": [75.25,110.34],
    "rewards": False
}

print(customer["purchases"])
```

```
[75.25, 110.34]
```

`KeyError` exception

- If you try to use a key that is not in the dictionary, a `KeyError` is raised

```
inventory = {
    "cheese": 10,
    "milk": 37,
    "yogurt": 6
}

print(inventory["kefir"])

KeyError: 'kefir'
```

Check if a key is in a dictionary

- Use the keyword `in` to check if a key exists in a dictionary

```
inventory = {
    "cheese": 10,
    "milk": 37,
    "yogurt": 6
}

print("kefir" in inventory)
```

```
False
```

IT'S CODING TIME

- Create a student dictionary
- Print all the dictionary
- Let's find a student number

Modify dictionaries

Add items to a dictionary

- Add an item to a dictionary by assigning a value to a new key

```
inventory = {
    "cheese": 10,
    "milk": 37,
    "yogurt": 6
}

inventory["butter"] = 22
inventory["cream"] = 18
print(inventory)

{'cheese': 10, 'milk': 37, 'yogurt': 6, 'butter': 22, 'cream': 18}
```

Modify items in a dictionary

- Modify the value of an item in a dictionary by using its key

```
inventory = {
  "cheese": 10,
  "milk": 37,
  "yogurt": 6
}

inventory["cheese"] = 15
inventory["milk"] = inventory["milk"] - 5
print(inventory)
```

```
{'cheese': 15, 'milk': 32, 'yogurt': 6}
```

Examine the code

- What is the output of this code block?

```
basketball_team = {
    "Ariel": "Point guard",
    "Rosario": "Shooting guard",
    "Dana": "Small forward",
    "Kim": "Power forward",
}

basketball_team["Izumi"] = "Center"
print(basketball_team)
```

Examine the code

- What is the output of this code block?

```
basketball_team = {
    "Ariel": "Point guard",
    "Rosario": "Shooting guard",
    "Dana": "Small forward",
    "Kim": "Power forward",
}

basketball_team["Izumi"] = "Center"
print(basketball_team)
```

```
{'Ariel': 'Point guard', 'Rosario': 'Shooting guard', 'Dana': 'Small forward',
'Kim': 'Power forward', 'Izumi': 'Center'}
```

Remove items from a dictionary

- We can remove an item from a dictionary with .pop()

```
inventory = {
    "cheese": 10,
    "milk": 37,
    "yogurt": 6
}

inventory.pop("milk")
inventory.pop("yogurt")
print(inventory)
```

```
{'cheese': 10}
```

Avoid `KeyError` exceptions

- Use an `if` statement to check if an item is in a dictionary before attempting to remove the item

```
inventory = {
    "cheese": 10,
    "milk": 37,
    "yogurt": 6
}

def remove_item(inventory, item):
    if item in inventory:
        inventory.pop(item)
        print(f"Updated inventory: {inventory}")
    else:
        print("Cannot remove item that isn't in inventory")

remove_item(inventory, "kefir")
```

```
Cannot remove item that isn't in inventory
```

Check your understanding

- What is the output of this code block?

```
car = {
    "make": "Chevrolet",
    "model": "Corvette",
    "year": 1963,
    "mileage": 100000
}

car["model"] = "Camaro"
car["year"] = 1969
car.pop("mileage")
print(car)
```

Check your understanding

- What is the output of this code block?

```
car = {
    "make": "Chevrolet",
    "model": "Corvette",
    "year": 1963,
    "mileage": 100000
}

car["model"] = "Camaro"
car["year"] = 1969
car.pop("mileage")
print(car)
```

```
{'make': 'Chevrolet', 'model': 'Camaro', 'year': 1969}
```

IT'S CODING TIME

- Add major to student dictionary
- Remove address from student dictionary
- Creating an upr email for a student

Iterate through dictionaries

Iterate through a dictionary to display its keys

```
inventory = {
    "cheese": 10,
    "milk": 37,
    "yogurt": 6
}

for product in inventory:
    print(product)              # Print loop variable to display keys
```

```
cheese
milk
yogurt
```

Examine the code

- What is the output of this code block?

```
capitals = {
    "Kenya": "Nairobi",
    "Peru": "Lima",
    "Japan": "Tokyo"
}

for country in capitals:
    print(country)
```

Examine the code

- What is the output of this code block?

```
capitals = {
    "Kenya": "Nairobi",
    "Peru": "Lima",
    "Japan": "Tokyo"
}

for country in capitals:
    print(country)
```

```
Kenya
Peru
Japan
```

Iterate through a dictionary to display its values

```
inventory = {
    "cheese": 10,
    "milk": 37,
    "yogurt": 6
}

for product in inventory:
    print(inventory[product])      # Place the loop variable in brackets
```

```
10
37
6
```

Check your understanding

- What code should you add to the highlighted section to display all the values?

```
product_data = {
    "name": "Coffee Maker",
    "price": 99.99,
    "category": "Appliances",
    "in_stock": True
}

for data in product_data:
    print(???)
```

Check your understanding

- What code should you add to the highlighted section to display all the values ?

```
product_data = {
    "name": "Coffee Maker",
    "price": 99.99,
    "category": "Appliances",
    "in_stock": True
}

for data in product_data:
    print(product_data[data])
```

```
Coffee Maker
99.99
Appliances
True
```

Iterate through a dictionary to manipulate its data

```
inventory = {
    "cheese": 10,
    "milk": 37,
    "yogurt": 6
}

for product in inventory:
    inventory[product] *= 4        # Updates a value by multiplying it by 4
    print(inventory[product])
```

```
40
148
24
```

IT'S CODING TIME

- Add a student the a dictionary of students the key is an email and the value is student information

Why are dictionaries important?

- They are very fast for lookups
- They’re flexible and easy to modify
- They’re everywhere in real-world programs

Questions?

Post to our Discord server

or

email:

upr-ciic3015-staff@upr.edu

karelys.lopez1@upr.edu

Give us anonymous feedback!

https://bit.ly/CIIC3015_F25_Feedback

CODE PATH .ORG

Sets

Introduction: Sets

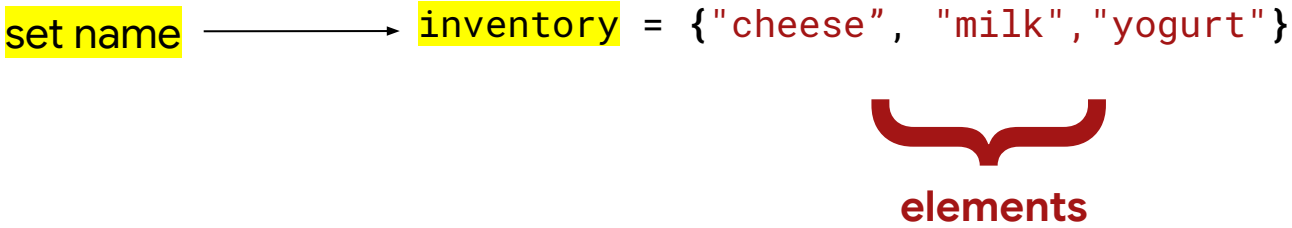
- A **set** is an **unordered** collection of unique items.
  - Sets store individual values, not pairs like dictionaries.
  - Each element in a set is distinct
  - Sets are mutable like dictionaries and lists
  - You can combine data without repeats.

```
inventory = {"cheese", "milk", "yogurt"}
```

Syntax: Create a set

The general syntax for creating a set:

1. Start with the set name followed by the assignment operator (=)
2. Use curly braces ({}) to enclose the contents of the set
3. Use commas (,) to separate elements from each other



Syntax: Create a set, another way

The general syntax for creating a set:

1. Start with the set name followed by the assignment operator (=)
2. Use set () to initiate a set

set name ⟶

```
items = set(['couch', 'pillow', 'couch'])
```

```
items will be {'couch', 'pillow'}
```

Empty sets

```
items = set() ✅
```

```
items = {} ❌
```

This will create a dictionary and not a set

Operation “in”

`in` operator: check whether value exists in set

On average runs in **much faster** than in a list.

```
years = {1995, 2020, 1992, 1920, 1776}
1995 in years                          True
2019 in years                          False
```

Add items to set

- Add an item to a set

```
inventory = {"cheese", "milk", "yogurt"}

inventory.add("cream")
inventory.add("butter")

print(inventory)

{'cheese', 'milk', 'yogurt', 'cream', 'butter'}
```

Delete items in a set

- Delete a specific item in a set

```
inventory = {"cheese", "milk", "yogurt", “cream”, “butter”}

inventory.remove("cream")
inventory.remove("butter")

print(inventory)

{'cheese', 'milk', 'yogurt'}
```

`KeyError` exception

- If you try to delete an element that is not in the set, a `KeyError` is raised

```
inventory = {"cheese", "milk", "yogurt"}

inventory.remove("butter")

KeyError: 'butter'
```

Delete items in a set

- Delete a random item in a set

```
inventory = {"cheese", "milk", "yogurt", "cream", "butter"}

inventory.pop()
inventory.pop()

print(inventory)
```

```
{'milk', 'cheese', 'butter'}
```

Check your understanding

- What happens in this code?
- How can we avoid KeyError?

```
products = {"books", "pencil", "book", "notebook"}

products.remove("backpack")
```

```
KeyError: 'backpack'
```

Check your understanding

- What happens in this code?
- How can we avoid KeyError?

```
products = {"books", "pencil", "book", "notebook"}

if "backpack" in products:
     products.remove("backpack")

else:
     print(“Woops! that is not in the products set”)
```

```
Woops! that is not in the products set
```

Iterate through a set

```
inventory = {"cheese", "milk", "yogurt"}

for product in inventory:
    print(product)
```

What will it print?

It can print this…	or this…	or this!
cheese milk yogurt	yogurt cheese milk	milk yogurt cheese

No guaranteed order!

IT'S CODING TIME

- Create an empty set
- Add elements (Including the same element multiple times)
- Delete a specific element
- Delete a random element
- Iterate over a set
- Find how many elements many unique characters in a string

Recap

Sets

- A **set** is an **unordered** collection of unique items.
  - Each element in a set is unique
  - Sets are mutable like dictionaries and lists
  - You can combine data without repeats.
  - Iterating in the set is like we do in lists
  - We remove using .remove(element)
  - Adding to a set, use .add(element)

```
inventory = {"cheese", "milk", "yogurt"}
```

Math with sets

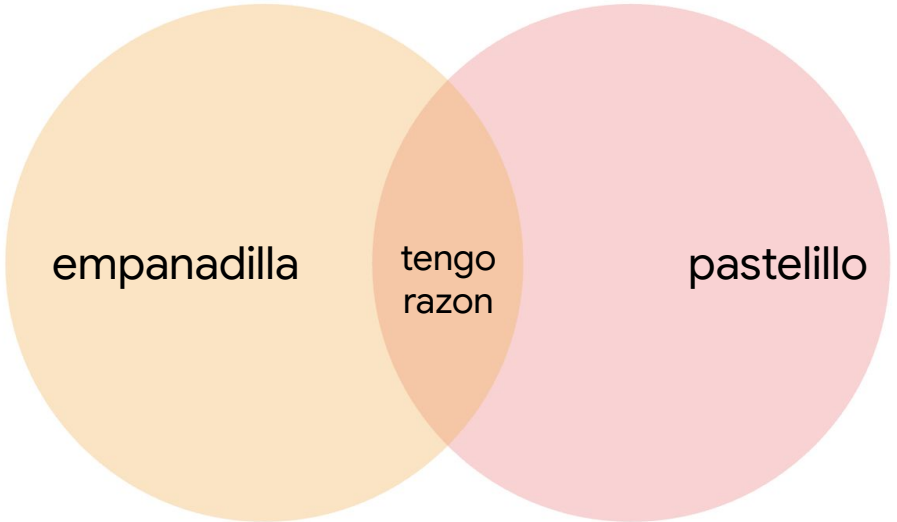
Set Math

- `s1.union(s2)` -> Combines two sets
- `s1.intersect(s2)` -> Finds overlap between two sets
- `s1.difference(s2)` -> Removes all items in s2 from s1

Set Math: Union

```
san_juan = {'empanadilla', 'tengo
razon'}
ponce = {'pastelillo', 'tengo razon'}
print(san_juan.union(ponce))
```

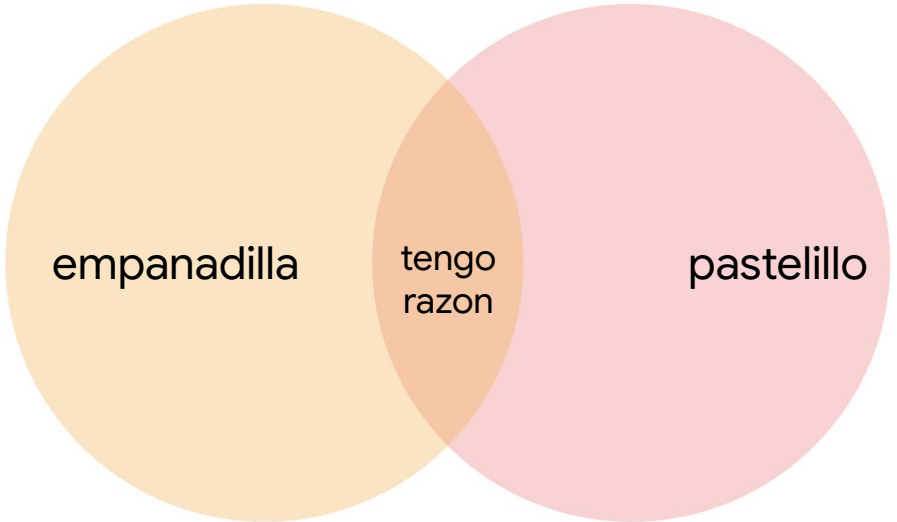
```
{'empanadilla', 'tengo razon',
pastelillo}
```



Set Math: Intersection

```
san_juan = {'empanadilla', 'tengo
razon'}
ponce = {'pastelillo', 'tengo razon'}
print(san_juan.intersection(ponce))
```

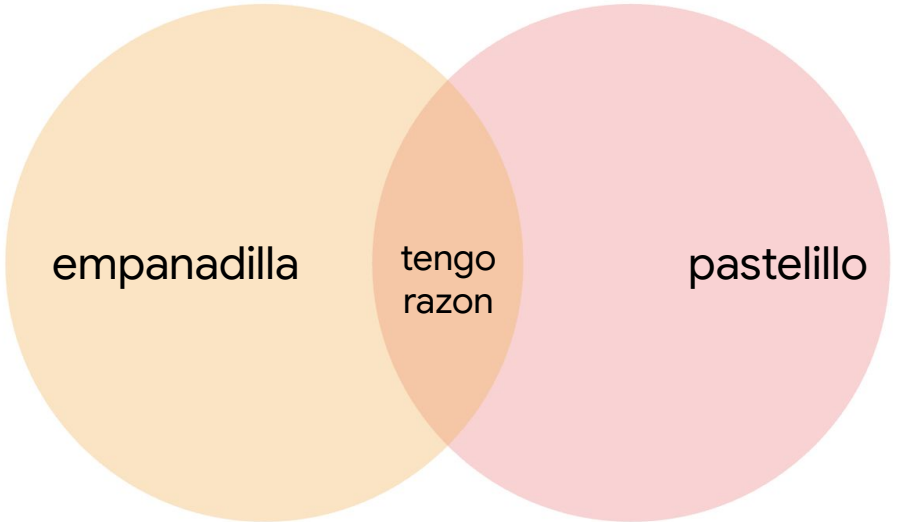
```
{'tengo razon'}
```



Set Math: Difference

```
san_juan = {'empanadilla', 'tengo
razon'}
ponce = {'pastelillo', 'tengo razon'}
print(san_juan.difference(ponce))
```

```
{'empanadilla'}
```



Questions?

Post to our Discord server

or

email:

upr-ciic3015-staff@upr.edu

karelys.lopez1@upr.edu

Give us anonymous feedback!

https://bit.ly/CIIC3015_F25_Feedback

CODE PATH .ORG